Софийски университет „Св. Климент Охридски“

Мултимедийни уеб технологии с Angular и Typescript

Документация

Система за управление на задания по проекти

Изготвил: Тодор Борисов Борисов

Факултетен номер: 81323 дата: 28.06.2019 г.

1.Функционални и нефункционални изисквания

1.1 Фунцкионални изисквания:

* Възможност за работа на администраторски потребител и на обикновен потребител
* Възможност за създаване, редактиране и изтриване на потребители от страна на администратора
* Възможност за създаване, редактиране и изтриване на проекти и назначаване за работа върху такива на потребителни от страна на администратора
* Възможност за създаване, редактиране и изтриване на срещи, които са част от конкретен проект от страна на администратора
* Възможност за създаване, редактиране и изтриване на задания, които са част от конкретна среща от страна на администратора
* Възможност за редактиране на задание конкретнен проект и конкретна негова среща от назначен от администратора за този проект потребител

1.2 Нефункционални изисквания:

* Лекота на упортреба и интуитивен интерфейс – потребителят трябва да може да работи пълноценно със системата с минимално ръководство
* Стабилност – системата не трябва да води до забиване на клиента, или на сървъра, на който е инсталирана
* Брой потребители – системата трябва да може да поддържа множество потребителни
* Бърз достъп –търсенето в системата трябва да е достатъчно бързо

2. Използвани технологии

2.1 Frontend

За изграждането на frontend-a е използвана технологията Angular2+, базирана на TypeScript

Използвани библиотеки:

* PrimeNg - https://www.primefaces.org/primeng/#/
* Angular Material - https://material.angular.io/

2.2 Backend –

За изграждането на backend-а са използвани технологиите Java(Enterprise Edition), Payara(Application Server) и Maven(инструмент за автоматизирано build-ване на проекти)

Използвани библиотеки:

* JUnit - https://junit.org/junit5/
* EclipseLink - https://www.eclipse.org/eclipselink/
* Javaee API - https://mvnrepository.com/artifact/javax/javaee-api/8.0
* Javax persistence -https://docs.oracle.com/javaee/7/api/javax/persistence/package-summary.html
* Jersey - https://jersey.github.io/

2.3 База данни – за база данни е използвана PostgreSQL

3. Архитектура, реализация и взаимодействие между основните компоненти на системата

3.1 Frontend – Архитектурата на frontend-а е изградена на базата на две основни групи – сървиси за комуникация с backend-a чрез Rest API-та, и компоненти, образуващи четири основни групи за управление на данните – управление на данните за потребители, управление на данните за проекти, управление на данните за срещи и управление на данните за задания. Всеки един от тези модули е реализиран така, че да предоставя функционалности свързани с извеждане и манипулация на конкретни данни и по-точно извършващи операциите – създаване, редактиране и изтриване на информацията комуникирана с бекенда(както и допълнителната финкционалност за назначаване на проекти на конкретни потребители). Конкретните компоненти използвани за реализация са: NewUserLayoutComponent, NewProjectLayoutComponent, NewMeetingLayoutComponent и NewTaskLayoutComponent с помощни компоненти NewAssignLayoutComponent, NewMeetingProjectLayoutComponent, NewTaskProjectLayoutComponent и NewTaskMeetingLayoutComponent, за конкретните операции – създаване и редактиране, са използвани компонентите CreateUserComponent, EditUserComponent, CreateProjectComponent, EditProjectComponent , CreateMeetingComponent, EditMeetingComponent , CreateTaskComponent, EditTaskComponent. Сървисите отговарящи за основните операции за комуникация с бекенда са GetUsersService, GetProjectsService, GetMeetingsService и GetTasksService.

3.2 Backend – за изграждането на архитектурата на бекенда е използвана подобна схема. Основните групи на разделение са четири – ресурси – UserResources, ProjectResources, MeetingResources, TaskResources, които отговарят за комуникацията с frontend-а, реализирани “entity”-та – User, Project, Meeting, Task, които улеснят записването на данни в базата данни, сървиси – UserServices, ProjectServices, MeetingServices, TaskServices, които извършват самото взаимодействие с базата данни, и допълнителни класове обработващи данните приемани и изпращани към и от backend-а.

4.Описание на архитектурано значимите интерфейси

Взаимодействието между frontend-а и backend-а се осъществява с помощта на REST service API-та. Във frontend-а те се класифицират под 5 групи:

1. Get-users

* createUser(user: User){ return this.http.post(`TaskProg/resources/userResources/createUser`, user); } – за създаване на потребител
* getUsersFull(){ return this.http.get<User[]>(`TaskProg/resources/userResources/getUserListFull`); } – за получаване на всички потребители
* getUsersAssignFull(projectkey : string){ return this.http.get<User[]>(`TaskProg/resources/userResources/getUserListAssignFull/${projectkey}`); } – за получаване на всички потребители, които са назначени към проект със ключ съответващ на projectkey
* getUsersAssignFullAssigned(projectkey : string){ return this.http.get<User[]>(`TaskProg/resources/userResources/getUserListAssignFullAssigned/${projectkey}`); } – за получаване на всички потребители, които не са назначени към проект с ключ на проект съответсващ на projectkey
* editUser(user: User){ return this.http.post(`TaskProg/resources/userResources/editUser`, user); } – за редактиране на съответния потребител
* deleteUser(username: string){ return this.http.get(`TaskProg/resources/userResources/deleteUser/${username}`) } – за изтриване на конкретния потребител
* isUserWokringOnProject(username: string, projectKey: string){ return this.http.get<boolean>(`TaskProg/resources/userResources/isUserWorkingOnProject/${username}/${projectKey}`); } – за проверка дали сътоветния потребител е назначен да работи по контретния проект

1. Get-projects

* createProject(project: Project){ return this.http.post(`TaskProg/resources/projectResources/createProject`, project); } – за създаване на проект
* getProjects(){ return this.http.get<Project[]>(`TaskProg/resources/projectResources/getProjectList`); } – за получаване на всички съществуваши проекти
* getUserProjects(username: string){ return this.http.get<Project[]>(`TaskProg/resources/projectResources/getUserProjectList/${username}`); } – за получаване на всички проекти към които е назначен конкретния потребител
* editProject(project: Project, prevProjectKey: string){ return this.http.post(`TaskProg/resources/projectResources/editProject/${prevProjectKey}`, project); } – за редактиране на проект с пропърти projectkey съответстващо на prevProjectKey
* deleteProject(projectKey: string){ return this.http.get(`TaskProg/resources/projectResources/deleteProject/${projectKey}`) } – за изтриване на конкретния проект
* assignUsersToProject(projectKey : string, username: string){ return this.http.get(`TaskProg/resources/projectResources/assignUsersToProject/${projectKey}/${username}`); } – за назначение на потребител към конкретен проект

1. Get-meetings

* createMeeting(projectKey: string, meeting: Meeting){ return this.http.post(`TaskProg/resources/meetingResources/createMeeting/${projectKey}`, meeting); } – за създаване на среща
* getMeetings(projectKey: string){ return this.http.get<Meeting[]>(`TaskProg/resources/meetingResources/getMeetingList/${projectKey}`); } – за получаване на всички срещи на проект с ключ съответстващ на projectKey
* editMeeting(meeting: Meeting, prevProjectKey: string){ return this.http.post(`TaskProg/resources/meetingResources/editMeeting/${prevProjectKey}`, meeting); } – за редактиране на среща с ключ meeting.meetingKey и на проект с ключ съответстващ на prevProjectKey
* deleteMeeting(projectKey: string, meeting: Meeting){ return this.http.post(`TaskProg/resources/meetingResources/deleteMeeting/${projectKey}`, meeting); } – за изтриване на среща с ключ meeting.meetingKey и на проект с ключ съответстващ на prevProjectKey

1. Get-tasks

* createTask(projectKey: string, meetingtask: MeetingTask){ return this.http.post(`TaskProg/resources/taskResources/createTask/${projectKey}`, meetingtask); } – за създаване на задание към проект с ключ projectKey и среща с ключ meetingTask.meetingKey
* getTasks(projectKey: string, meetingKey: string){ return this.http.get<Task[]>(`TaskProg/resources/taskResources/getTaskList/${projectKey}/${meetingKey}`); } – за получаване на всички задания по проект и среща с ключове съответстващи на projectKey и meetingKey
* editTask(prevProjectKey: string, meetingtask: MeetingTask){ return this.http.post(`TaskProg/resources/taskResources/editTask/${prevProjectKey}`, meetingtask); } – за редактиране на задание с проект и среща с ключове съответстващи на projectKey и meetingTask.meetingKey
* deleteTask(projectKey: string, meetingtask: MeetingTask){ return this.http.post(`TaskProg/resources/taskResources/deleteTask/${projectKey}`, meetingtask); } - за изтриване на задание с проект и среща с ключове съответстващи на projectKey и meetingtask.meetingKey

1. Login

* getIfValidUser(loginName: string, password: string){ return this.http.get<UserLogin>(`TaskProg/resources/userResources/validation/${loginName}/${password}`) } – за валидиране на потребителя

5.Начин на инсталация и конфигуриране на проекта

За Frontend-a:

* Трябва да има инсталиран Node.js
* Кодов редактор(напр. Visual Code)
* (Опционално) инсталиране на Angular-CLI
* Импортване на проетка чрез File -> Open Folder… и избиране на директорията на frontend-a
* За стартиране – в cmd-то се изпълнява командата ng serve –proxy-config proxy.conf.json

За базата данни:

* Инсталиране на PostgreSQL база данни(помнете username и password, които задавате, необходими са по-късно)
* Създаване на съответните таблици ръчно, или чрез backup файлa(TaskProg.backup)

За бекенда:

* Среда за разработване(напр. Eclipse for JavaEE development)
* Инсталиране на Maven - За конфигурацията на проекта се изпълнява командата “mvn clean install” в директорията на backend-a
* Сваляне на Payara – за конфигурация в Eclipse се сваля tool за Payara през Help -> Install new software и поставяне на линкът(https://raw.githubusercontent.com/payara/Payara\_PatchedProjects/eclipse/payaratools/oxygen/development) в полето Work with (за повече информация https://blog.payara.fish/payara-tools-unlocks-eclipse-for-payara-5). При създаване на Payara App Server се избира Payara от опциите, а за конфигурирането се избира домейнът, който се намира на ~\payara\payara5\glassfish\domains\domain1
  + В ~\payara\bin се изпълняват командите - asadmin create-jdbc-connection-pool --restype javax.sql.DataSource --datasourceclassname org.postgresql.ds.PGSimpleDataSource --property user=postgres:password=admin:serverName=localhost:databaseName=TaskProg:portNumber=5432 TPpool

// user и password трябва да бъдат същите, които са

// определени при инсталирането на базата данни

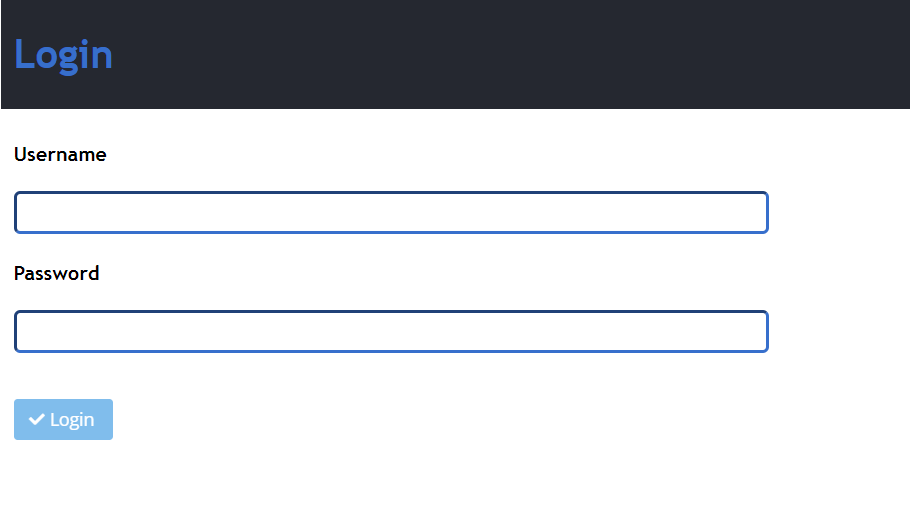
* + asadmin create-jdbc-resource --connectionpoolid TPpool jdbc/TPDB

* За да качите проекта в workspace-a на Eclipse, отидете на File -> Open projects from file system… -> Изберете директорията на backend-а
* Стартирайте Payara сървъра, след което с десен клик върху него отивате на Add and Remove… и пренасяте проекта в дясната половина

Всички bin директории на инсталирани ресурси трябва да бъдат включени към Environmental Variables

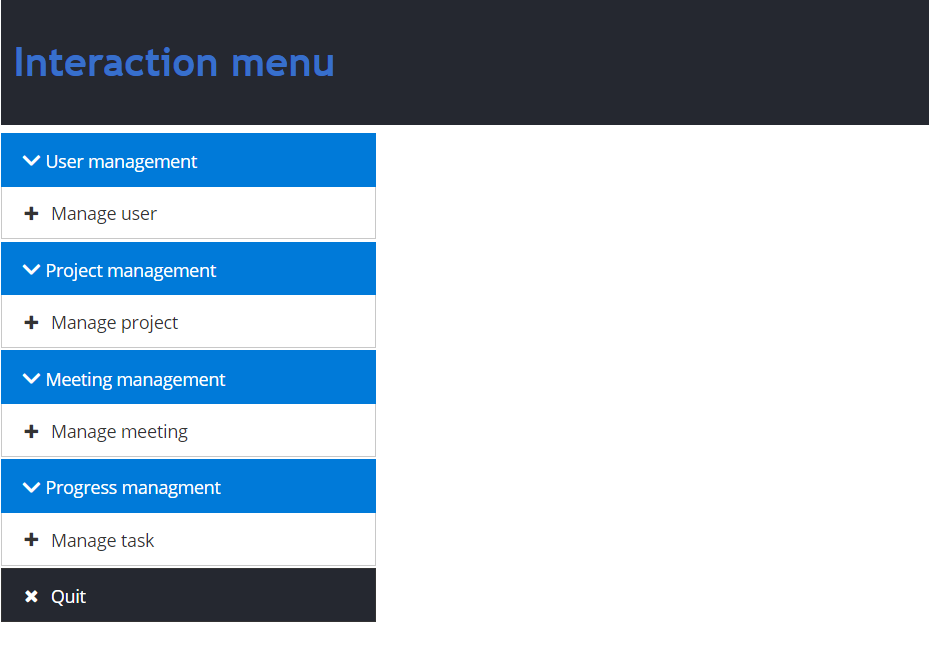
6. Потребителска документация за работа със системата –

Логин:

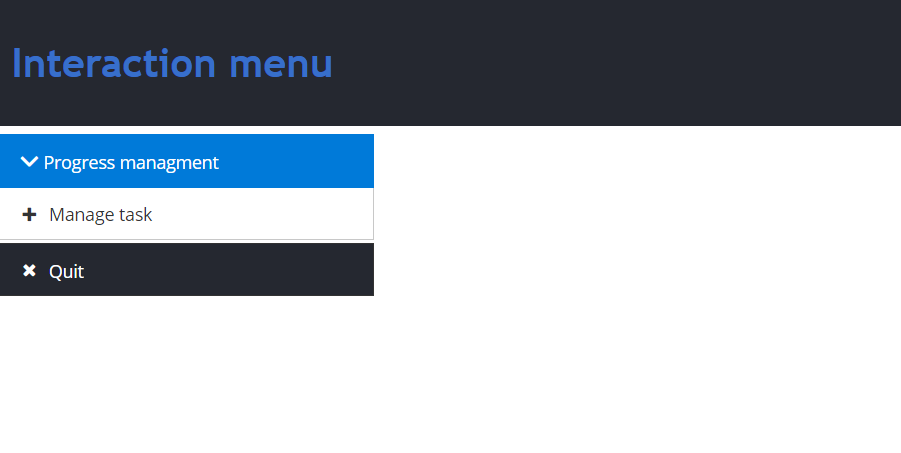


В зависимост от това, каква е ролята на потребителя(админ или обикновен потребител) се определят фукционалностите на интерактивното меню в началната страница:

* За Админ

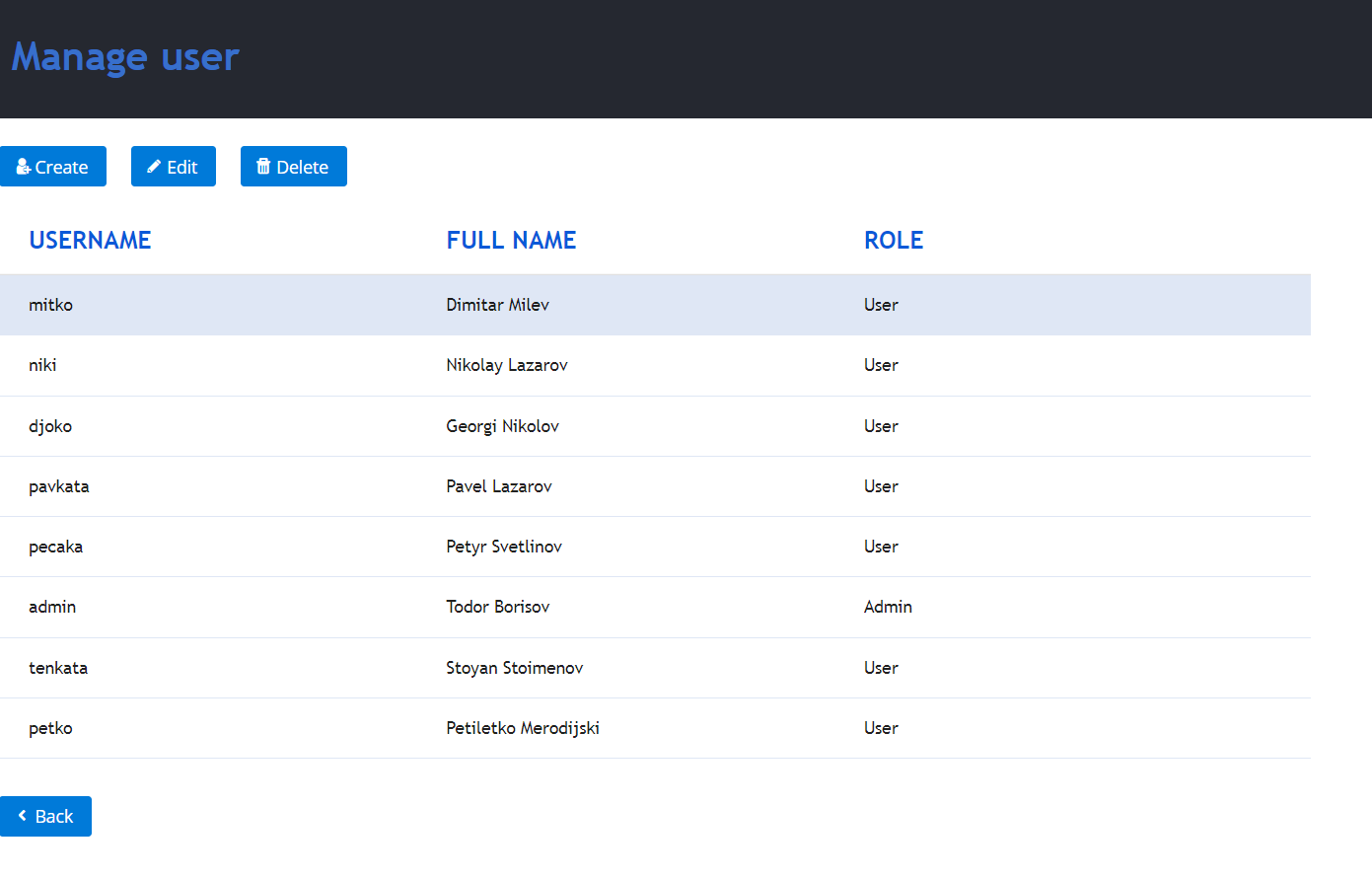


* За обикновен потребител

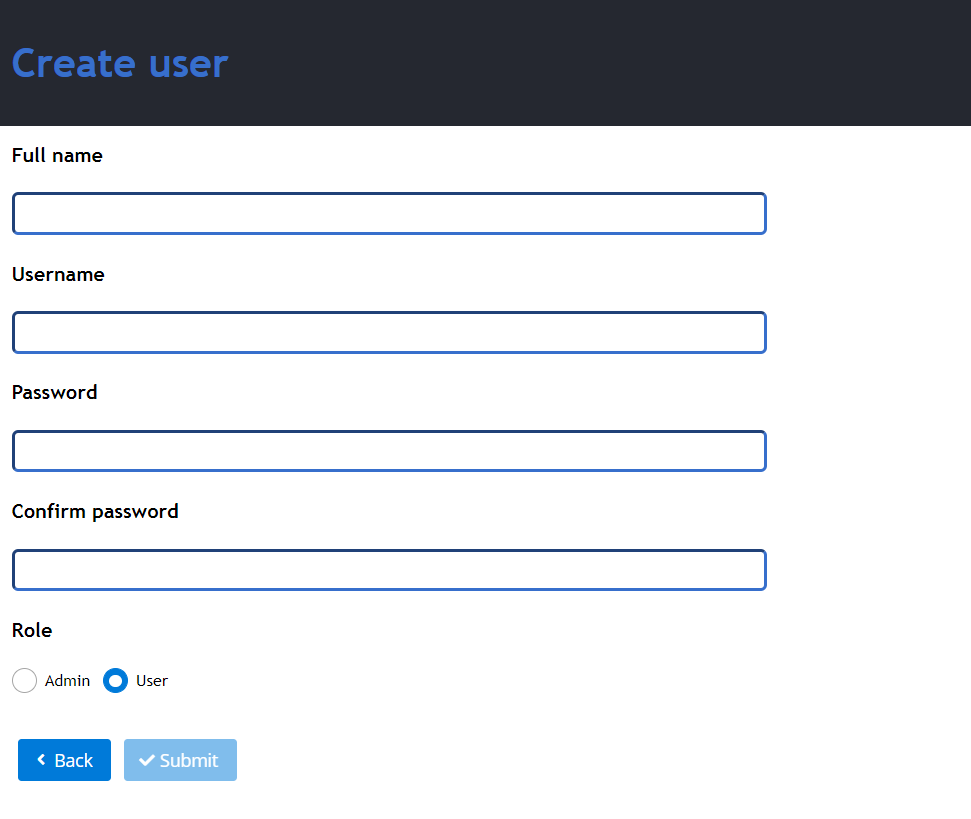


Функционалността на обикновения потребител е ограничена само за редактиране на задания на срещи по проекти, в които е включен, докато админа има достъп до всички фунцкионалности и затова ще разгледаме неговите

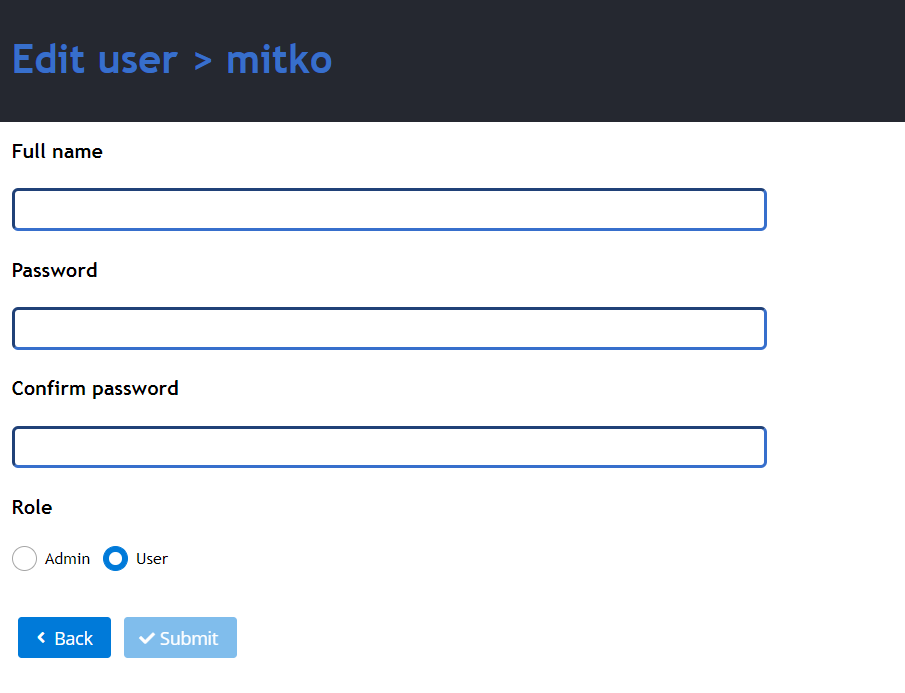
* Управление на потребители



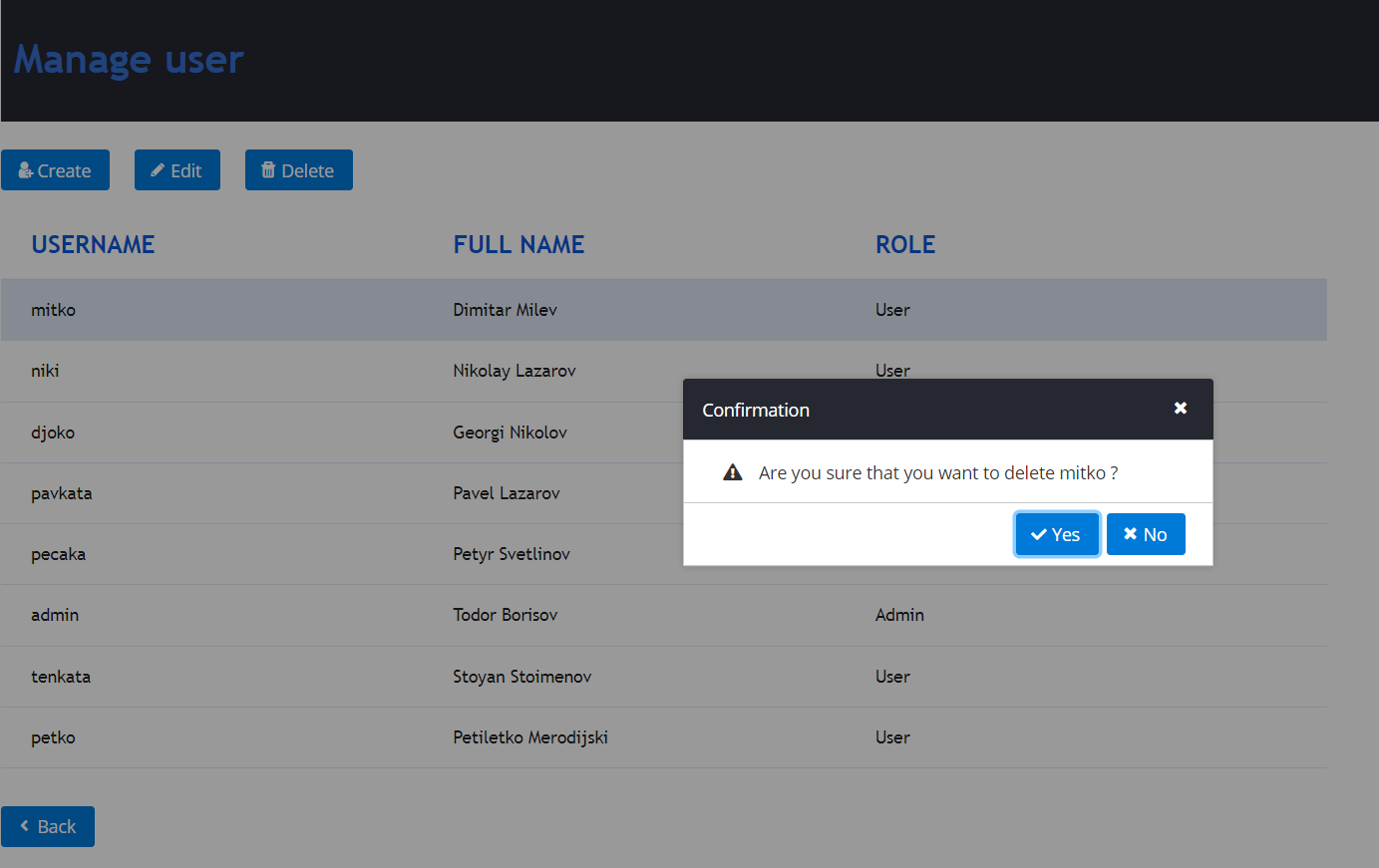
* Създаване



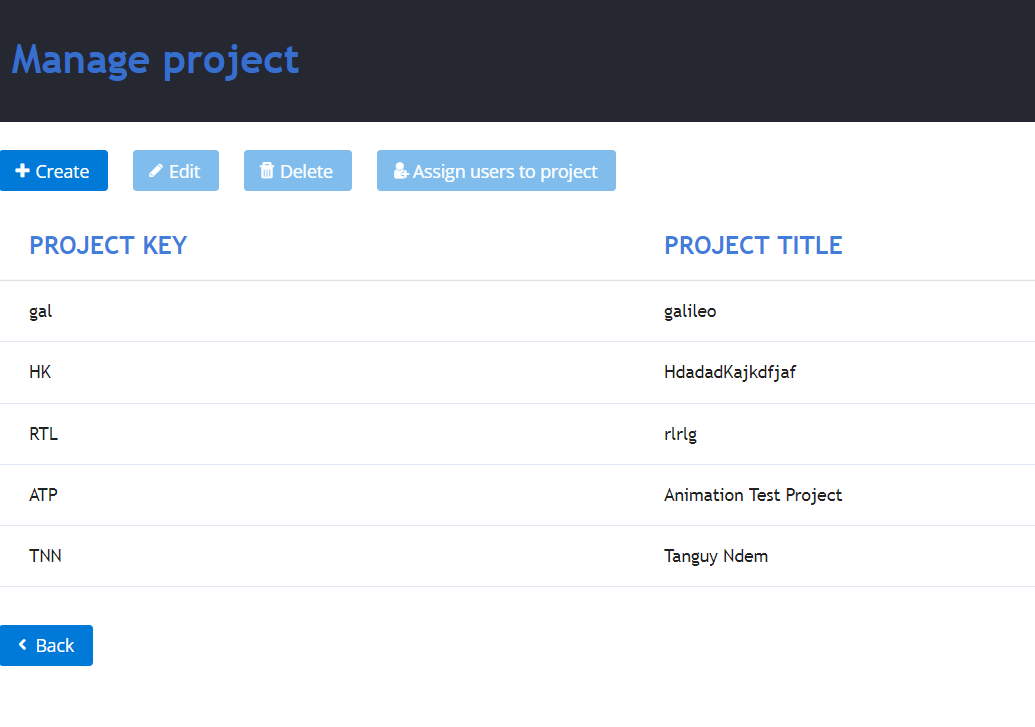
* Редактиране



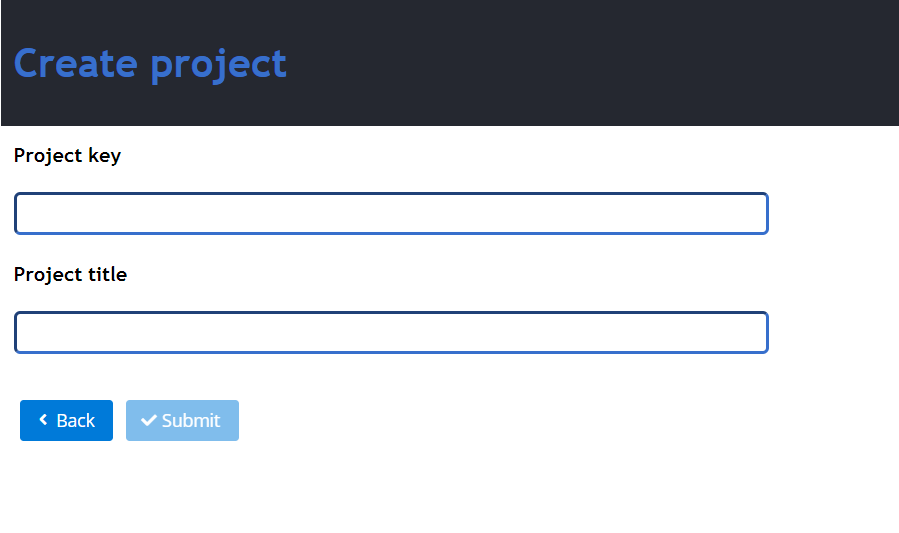
* Изтриване



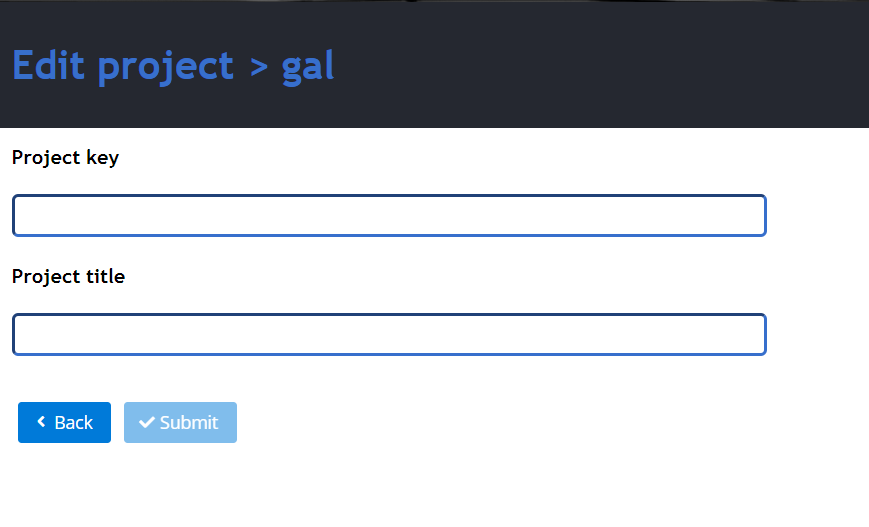
* Управление на проекти



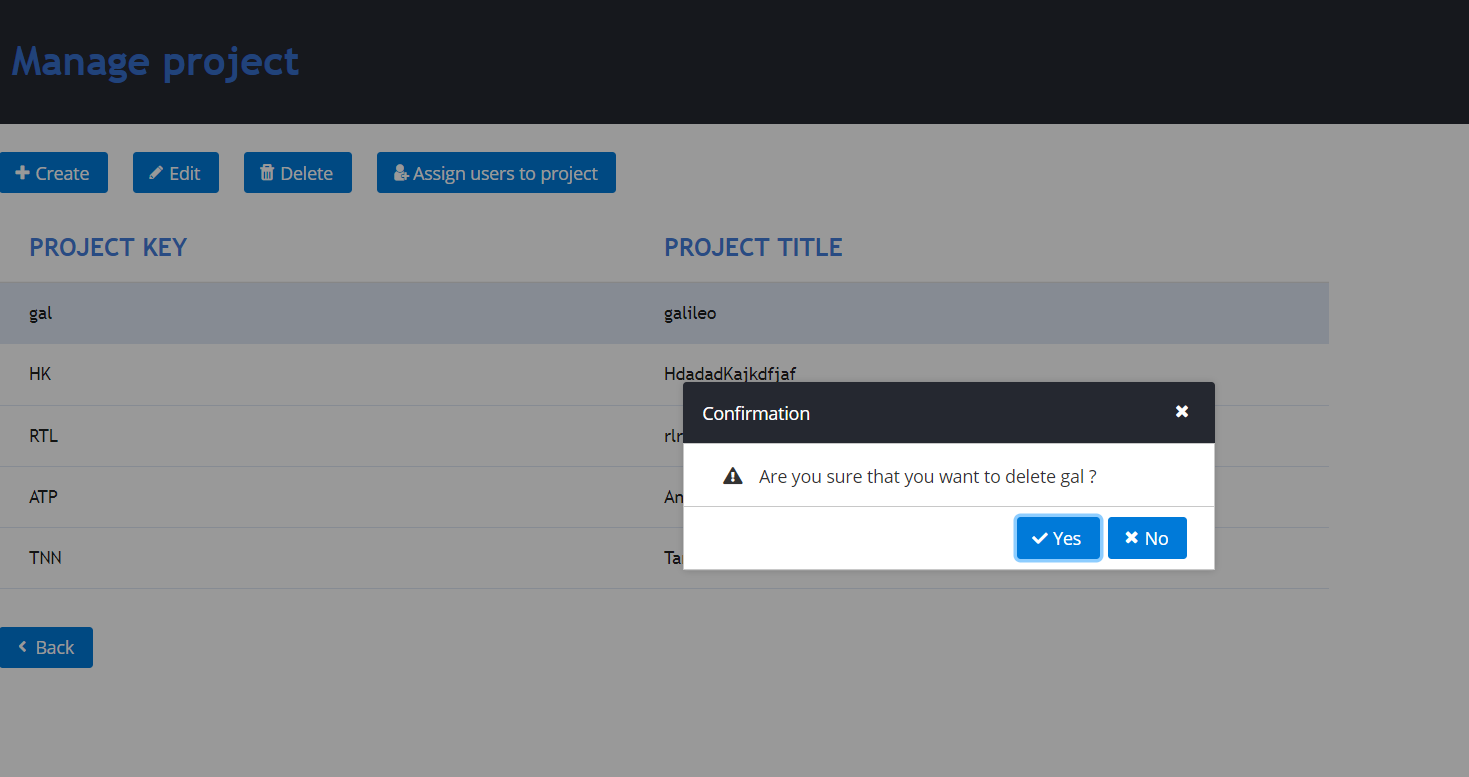
* Създаване на проект



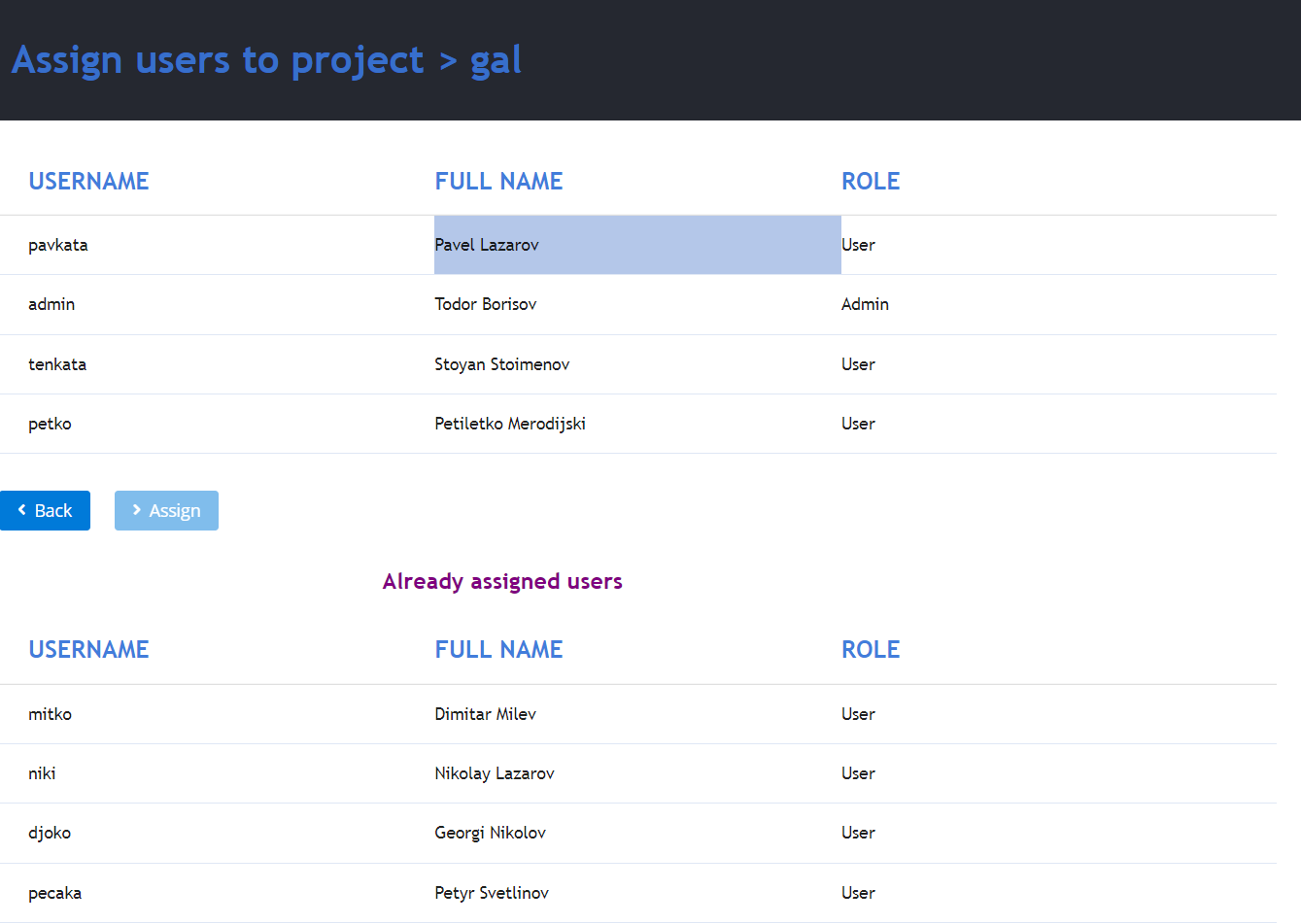
* Редактиране на проект



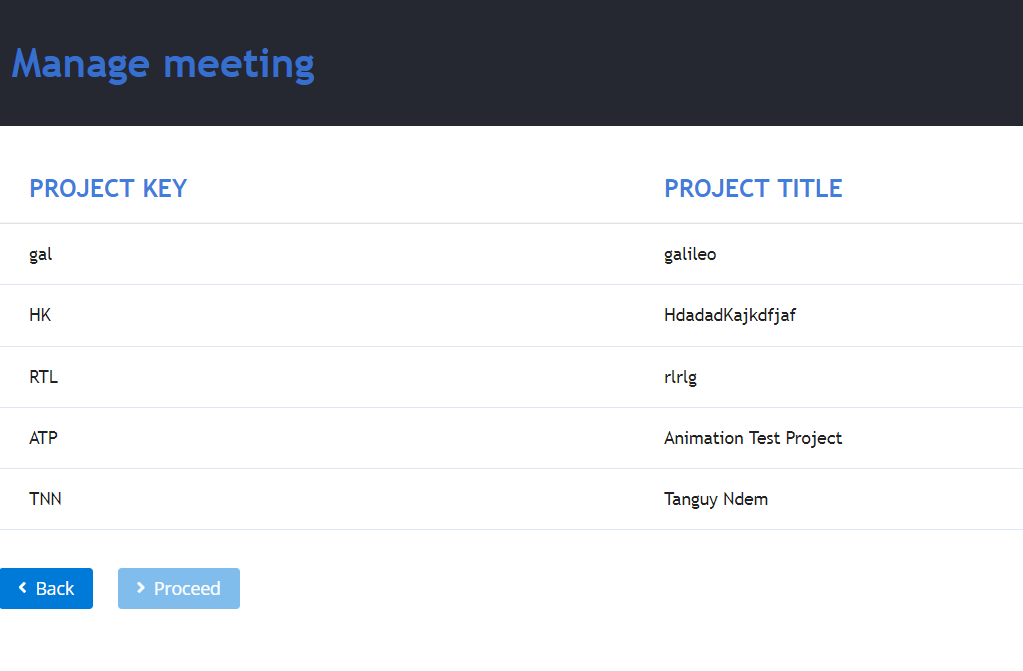
* Изтриване на проект

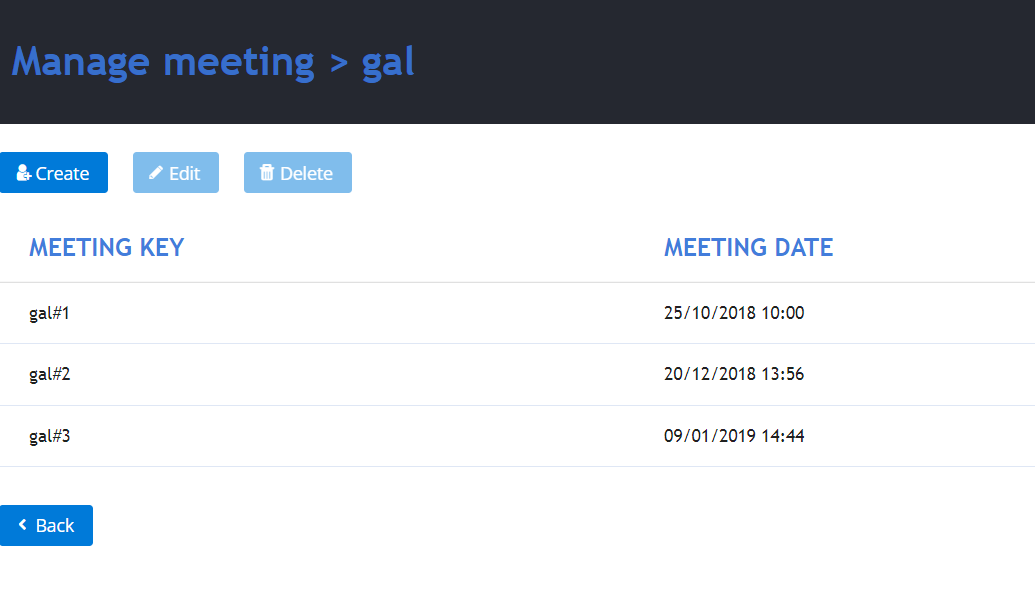


* Назначаване на проект на потребител

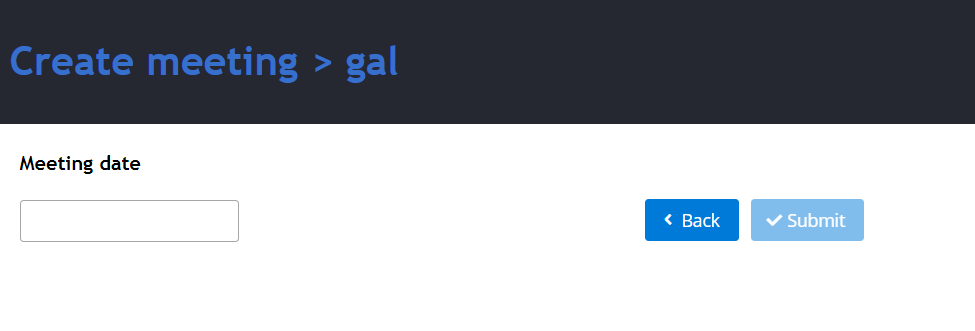


* Управление на срещи

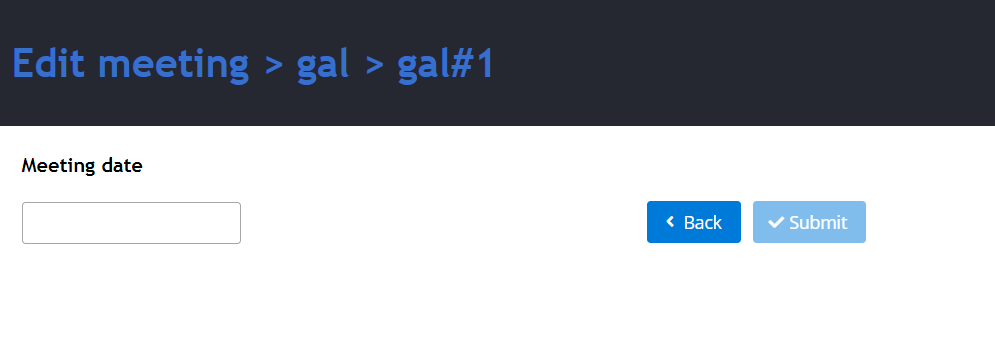




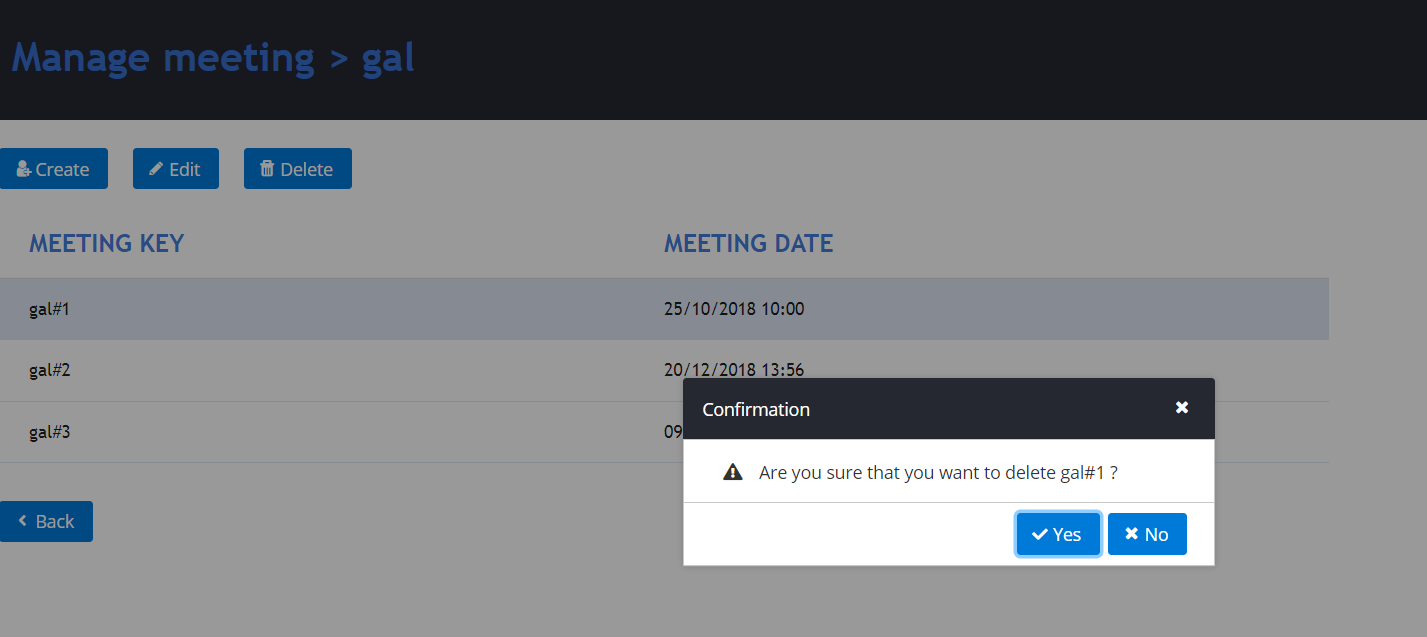
* Създаване на среща



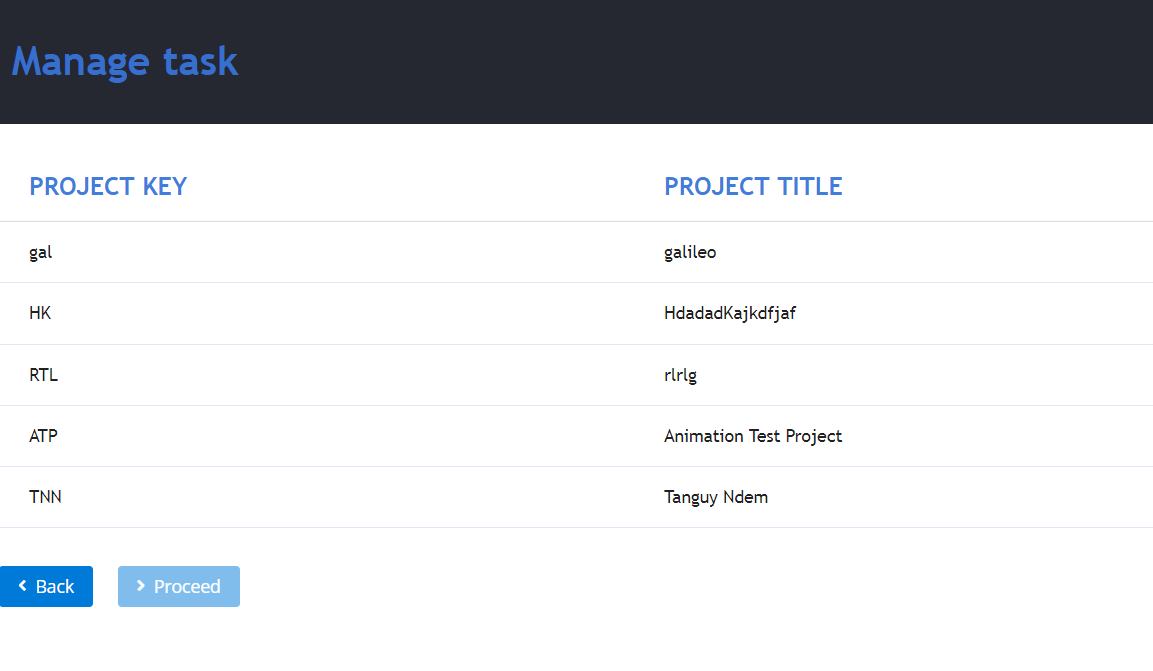
* Редактиране на среща

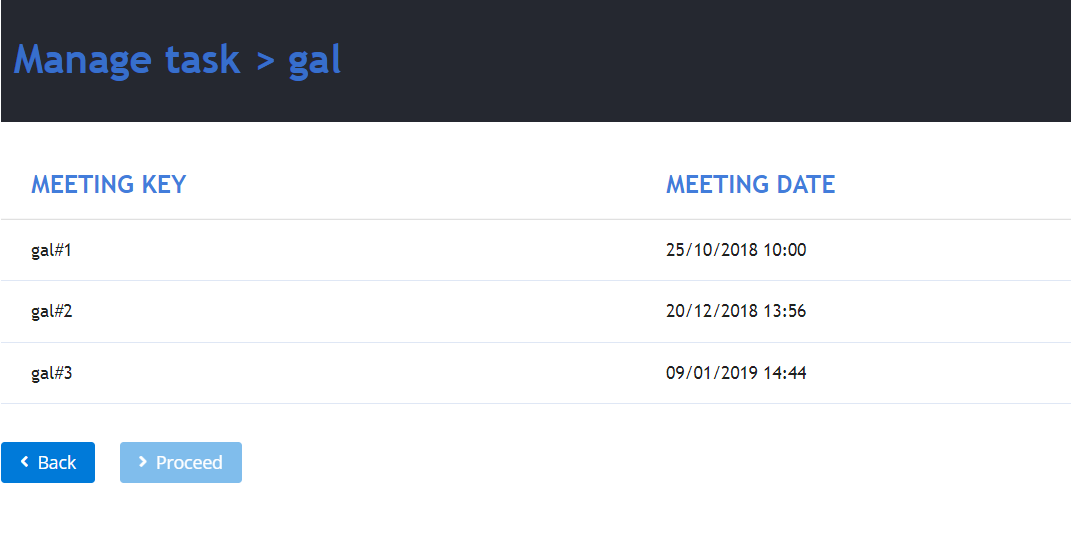


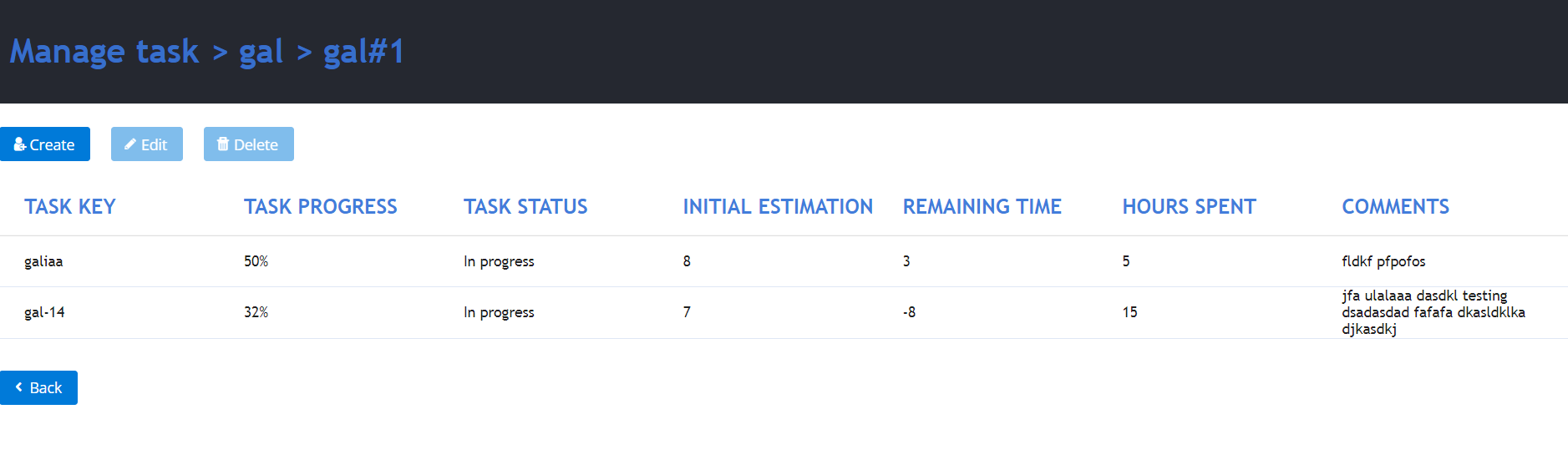
* Изтриване на среща



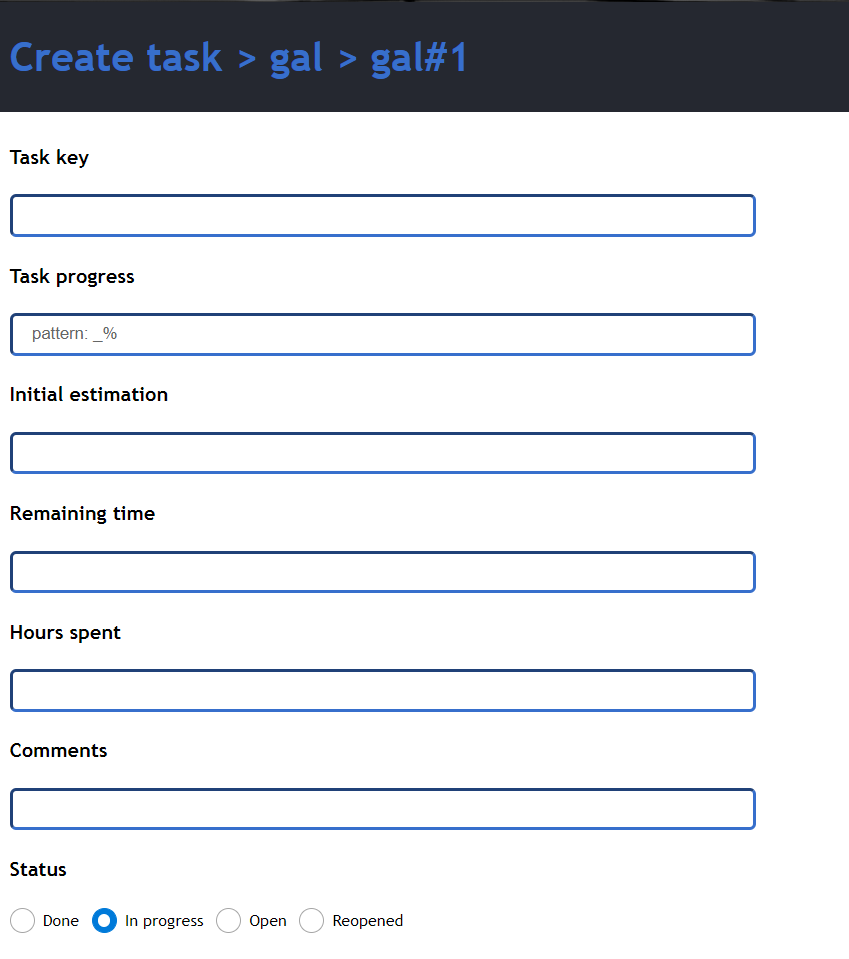
* Управление на задания



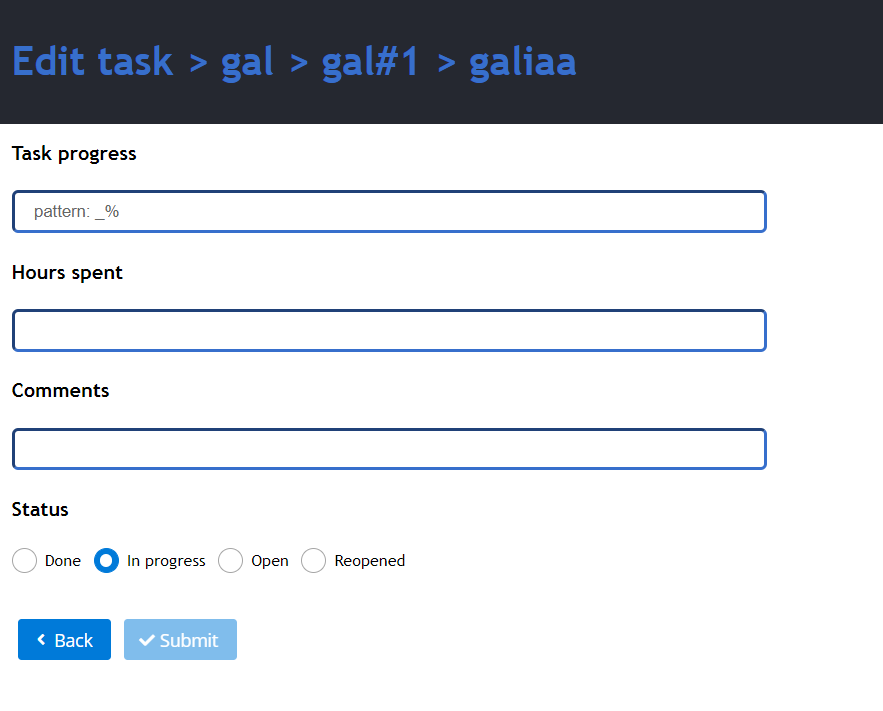




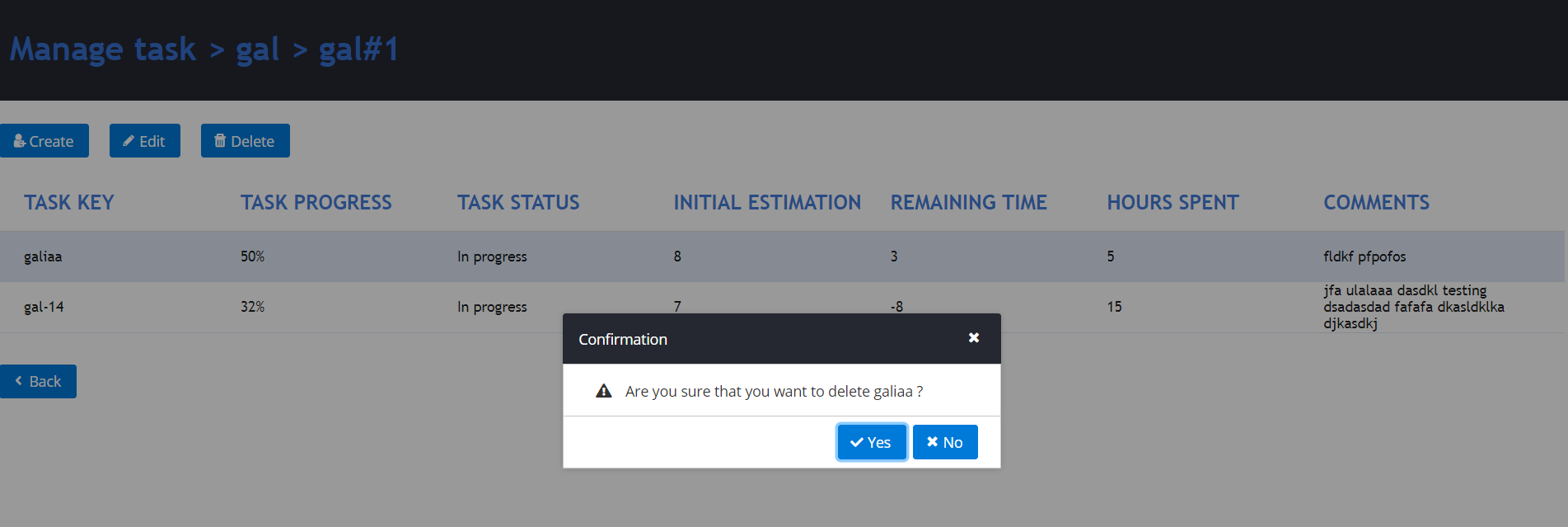
* Създаване на задание



* Редактиране на задание



* Изтриване на задание



7.Заключение

Не се бях занимавал с Angular, преди началото на този курс, и изграждането на конкретния проект ми предостави възможността да опозная тази технология и да я интегрирам и в реален проект. Смятам за много полезно, че обогатих знанията си по програмиране с конкретния application framework и бих продължил с допълнително обучение в тази насока. Относно конкретния проект, бих казал, че с него положих основите на едно по-обширно приложение, в което, в бъдеще бих искал да включа функционалности като Мейлинг система, Система позволяваща изграждането на статистики и изобразяването им графически.

8.Използвани ресурси

Getting started with Angular – Second Edition – Minko Gechev

Документацията на Angular - https://angular.io/